intel®

AP-709

APPLICATION NOTE

Maximizing Performance Using the MCS® 251 Microcontroller— Programming the 8XC251SB

SEONG FOOK TSEN
TECHNICAL MARKETING ENGINEER
EIGHT BIT MICROCONTROLLERS

April 1995

Order Number: 272671-001

Maximizing Performance Using the MCS® 251 Microcontroller—Programming the 8XC251SB

CONTENTS PAGE

1.0 INTRODUCTION

The MCS® 251 microcontroller is Intel's next generation MCS 51 microcontrollers. The MCS 251 microcontroller's first product, the 8XC251SB, is binary code and pin compatible with the existing 80C51 microcontrollers and its derivatives. With instruction pipeline and register based architecture, the 8XC251SB CPU executes most of the instructions in 1 state or 2 clocks period compared with 6 states or 12 clocks period in MCS 51 microcontrollers. Larger and more flexible memory spaces are available with its 24-bit linear memory addressing capability. 8XC251SB has more programming capability. The instruction set has been enriched to provide 16-bit and 32-bit capability. 8XC251SB has more bit addressable space. It has more addressing modes, register to register operations and improved data manipulation, accessing, and transfer capabilities. More control instructions are also available to optimize program flow.

This application note will describe the programming differences between MCS 51 microcontrollers and the 8XC251SB microcontrollers. The programming differences showed are memory organization, programming registers, addressing mode and instruction sets. Examples of how application codes written for MCS 51 microcontrollers can be optimized using 8XC251SB will also be provided in the following chapters and appendixes section. The state time calculation for the 8XC251SB in the examples are based on the 8XC251SB in source mode and the codes are executed from internal memory.

2.0 8XC251SB CPU FEATURES

The following is a list of major features of the 8XC251SB CPU.

- Fully compatible with the MCS 51 microcontroller's CPU
- Fully binary and source code compatible with the MCS 51 microcontroller's CPU
- Pipeline CPU architecture
- Register based architecture
- Linear address space
- 128 Kbyte external memory space (not available in MCS 51 microcontrollers)
- Additional addressing modes (not available in MCS 51 microcontrollers)
- Enriched instruction set
- Faster instruction execution time
- 8-, 16- and 32-bit data transfer instructions
- 8-, 16- and 32-bit control instructions
- 8-, 16- and 32-bit arithmetic instructions
- 2 bytes/state code fetch (internal execution)
- 2 byte/state data transfer (internal execution)

3.0 MEMORY ORGANIZATION

The 8XC251SB has three separate address spaces: 24-bit linear memory address space, Special Function Registers (SFRs) and a Register File.

3.1 Code and Data Memory Space

MCS® 51 microcontrollers have two separate 64 Kbyte code and data memory address space. Unlike MCS 51 microcontrollers, the 8XC251SB has a linear memory address space, that is, unsegmented. The 8XC251SB implements 4 of the 256 possible MCS 251 microcontroller's memory regions and they are (00:), (01:), (FE:) and (FF:) regions. The program code and data can reside within the same memory region unless the region has been specifically reserved for either code or data space. The 83C251SB/87C251SB (ROM/OTP) have 16 Kbytes of internal code memory that is mapped to location FF:0000H to FF:3FFFH of the linear address space. The 8XC251SB has 1 Kbyte of internal data memory that is mapped to location 00:0000 to 00:041FH. Refer to Figure I. For compatibility reasons, the MCS 51 microcontroller's program code and data memory address space can be mapped directly into the 8XC251SB linear memory address space.

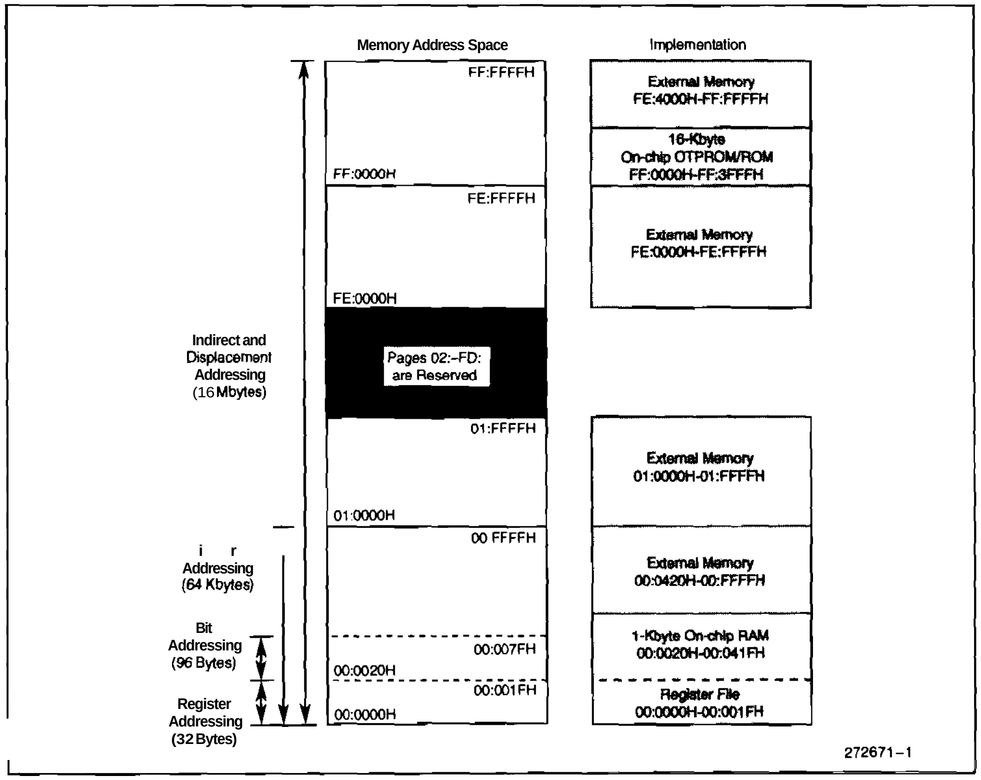
Memory Address Space
Implementation
FF:FFFFH
FF:0000H
FE:FFFFH
FE:0000H
Pages 02:-FD: are Reserved
01:FFFFH
01:0000H
00 FFFFH
00:007FH
00:0020H
00:001FH
00:0000H
External Memory FE:4000H-FF:FFFFH
16-Kbyte On-chip OTPROM/ROM FF:0000H-FF:3FFFH
External Memory FE:0000H-FE:FFFFH
External Memory 01:0000H-01:FFFFH
External Memory 00:0420H-00:FFFFH
1-Kbyte On-chip RAM 00:0020H-00:041FH
Register File 00:0000H-00:001FH
Indirect and Displacement Addressing (16 Mbytes)
i r Addressing (64 Kbytes)
Bit Addressing (96 Bytes)
Register Addressing (32 Bytes)
272671-1

Figure 1. 8XC251SB Memory Address Space

Many applications today demand for more code and data memory in the current microcontrollers. The 8XC251SB resolves the memory limitation by providing more code and data memory. With linear memory address space capability which is unsegmented, application programs can have the flexibility to mix code and data. The 17 address lines (A0–A16), with RD# configured as A16, 8XC251SB can have up to 128 Kbytes of external code and data memory. It is possible to get a total of 145 Kbytes maximum of code and data memory space mixture, which can be formed using 1 Kbyte of internal data memory from region {00:}, 16 Kbytes of internal code memory from region {FF:} and 128 Kbytes of external code and data memory from region {01:} and {FE:}.

Examples 1a and 1b illustrate how applications can get a total 145 Kbytes of code and data memory.

```
; 8XC251SB to take advantage of the 128-Kbyte code and data memory.
: RD# configured as A16
; Main routine residing in the device internal code memory.

; External declaration
EXTRN     ECODE          (ROUTINE1, ROUTINE2)

DSEG      AT             00:0020H      ; On-chip data memory
{
          1-Kbyte of on-chip for data storage
}
CSEG      AT             OFF:0000H                ; Reset vector for 8XC251SB
          ljmp           MAIN

; Main program begin
; ==============
S_MAIN    SEGMENTCODE AT OFF:0100H
RSEG      S_MAIN
MAIN:
          |
          16-Kbyte of user code
          |
          ECALL          ROUTINE1  : Extended call to region
          {
          User code
          |
          ECALL          ROUTINE2  ; Extended call to region
          |
          User code
          |
END
```

Example 1a. Utilizing 16-Kbyte of Code and 1-Kbyte of Data Memory (On-Chip)

```
; 8XC251SB to take advantage of the 128-Kbyte code or data memory.
; RD# configured as A16
; Routine1 and Routine2 residing in the 8XC251SB external code memory space.

; Routine Declaration
PUBLIC ROUTINE1, ROUTINE2

S_ROUTINE1              SEGMENTCODE AT 0FE:0000H
RSEG       S_ROUTINE1
ROUTINE1:
           |
           64-Kbyte of user code or data
           |
           ERET

S_ROUTINE2              SEGMENTCODE AT 01:0000H
RSEG       S_ROUTINE2
ROUTINE2:
           |
           64-Kbyte of user code or data
           |
           ERET
END
```

Example Ib. Utilizing 128 Kbyte of External Code and Data

3.2 Register File

The 8XC251SB register file consist of 40 locations: 0-31 and 56-63 as shown in Figure 2. Register 0-15 can be accessed as byte register (Rn) addressing, word register (WRj) addressing or double-word register (Drk) addressing. Registers 16-31 can be accessed by word register addressing or double-word register addressing and registers 56-63 can be accessed by double-word register addressing. Register 32-55 are not available in the 8XC251SB. Note that register DR56 has been reserved for the extended Data Pointer, DPX, and DR60 has been reserved for the extended Stack Pointer, SPX. Registers DR56 and DR60 cannot be used as a general purpose register.

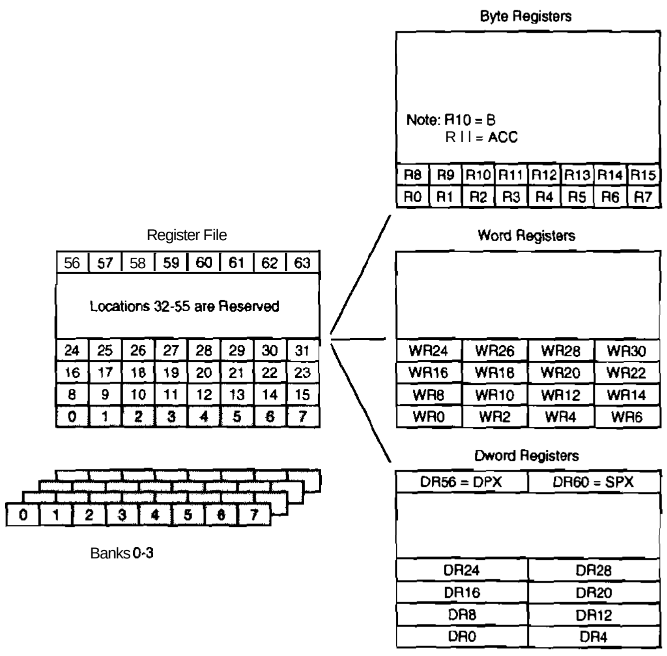

Figure 2. 8XC251SB Register File

The 8XC251SB has 32 additional registers that are not available in the MCS® 51 microcontrollers, these registers are register 8..31 and register 56..63. Applications will definitely benefit in software performance (that is reduction in execution time and code size) when using these registers for data manipulation. These registers provide advantage to application programs written in high-level programming language, such as 'C' language, because the compiler uses the registers to hold the variables in computing an expression. Furthermore, with the flexibility in using these registers as byte, word or double-word, applications code can take advantage of this feature.

Example 2a and 2b demonstrates the 8XC251SB register flexibility versus the MCS 51 microcontrollers accumulator based architecture to evaluate an arithmetic expression. Examples show that the 8XC251SB requires only 9 bytes of codes space versus MCS 51 microcontrollers that require 24 bytes of code space. Hence, there is code size reduction when the 8XC251SB enriched instructions set is used. Examples also clearly show that the application has become more efficient when the 8XC251SB is used. The 8XC251SB requires only 17 states time to execute the task versus MCS 51 microcontollers require 144 states time. Hence, this example shows that the 8XC251SB executes 8.4 times faster than MCS 51 microcontrollers. Also there is a code size reduction of about 30% when 8XC251SB is used.

```
; Using MCS 51 microcontroller to evaluate (V*W) - (X*Y)
; Assume R0=V, R1=W, R2=X and R3=Y
ORG         0000H
            ljmp       EVALUATE

ORG         0100H
EVALUATE:
; Evaluate (V*W)•
             mov       A, R0          ; Mov V into accumulator
             mov       B. R1          ; Mov W into B register
             mul       AB             ; (V*W)
             mov       R0, B          ; Store high byte result
             mov       R1, A          ; Store low byte result
; Evaluate (X*Y)
             mov       A, R2          ; Mov X into accumulator
             mov       B. R3          ; Mov Y into B register
             mul       AB             ; (X*Y)
             mov       R2, B          ; Store high byte result
             mov       R3, A          ; Store low byte result
; Evaluate 16-bit (V*W) - (X*Y)
             clr       C
             mov       A,R1           ; Move low byte of (V*W) into accumulator
             subb      A. R3          ; Low byte ((V*W) - (X*Y))
             mov       R1, A          ; Store low byte result ((V*W) - (X*Y))
             mov       A, R0          ; Move high byte of (V*W) into accumulator
             subb      A, R2          ; High byte ((V*W) - (X*Y))
             mov       R0, A          ; Store high byte result((V*W) - (X*Y))
END
; Total number of MCS 51 microcontroller states= 162
; Total number of MCS 51 microcontroller bytes= 24
```

Example 2a. Using MCS 51 Microcontrollersto Evaluate Arithmetic Expression

```
; Using 8XC251SB microcontroller to evaluate (V*W) - (X*Y)
: Assume R0=V, R1=W, R2=X and R3=Y
ORG         FF:0000H                  ; 8XC251SB reset vector
            ljmp       EVALUATE
ORG         FF:0100H
EVALUATE:
; Evaluate (V*W)
             mul       R0. R1         ; (V*W)
; Evaluate (X*Y)
             mul       R2, R3         ; (X*Y)
; Evaluate 16-bit (V*W)-(X*Y)
             sub       WR0. WR2       ; ((V*W)-(X*Y))
END
; Total number of 8XC251SB states= 12
; Total number of 8XC251SB bytes= 9
```

Example 2b. Using 8XC251SB to Evaluate Arithmetic Expression

Refer to Appendix B and Appendix C for more examples that use these registers

3.3 Special Function Register (SFR)

The MCS® 51 microcontroller SFRs (80H to FFH) are directly mapped into SFR space location S:80H to S:FFH. Hence these provide full compatibility for the SFRs that are available in the MCS 51 architecture. These SFRs are used for peripherals control, accumulator access, port access and etc.

The 8XC251SB has additional programming registers located in the SFR space that are not available in the MCS 51 architecture. These registers are DPX and SPX, and they will be explained in the Data Pointer and Stack Pointer section respectively. Refer to Table 1 for a comparison of programming registers between MCS 51 microcontrollers and the 8XC251SB.

Table 1. Programming Registers Comparison

Programming Registers	MCS® 51 Microcontrollers	8XC251SB
A (Accumulator)	X	X
B (Secondary Register)	X	X
SP (Stack Pointer)	X	X
SPX (Extended Stack Pointer)	—	X
DPTR (Data Pointer)	X	X
DPX (Extended Data Pointer)	—	X
PSW (First Program Status Word)	X	X
PSW1 (Second Program Status Word)	—	X
RO...R7	X	X
R8...R15	—	X
WRO...WR30	—	X
DR0..DR28, DR56, DR60	—	X

3.3.1 Program Counter (PC)

MCS® 51 microcontrollers have only 16-bit program counter that allows a maximum of 64 Kbytes of code memory. In comparison, the MCS 251 microcontroller has a 24-bit program counter that provides 16 Mbytes of code memory.

3.3.2 Program Status Word (PSW)

The 8XC251SB has two Program Status Words, PSW and PSWI, to reflect the current state of the CPU. The PSW is compatible with the PSW found in the MCS 51 microcontroller. PSW1 contains two new flags, Zero (Z) and Negative (N). The following flags are available in PSW: CY, AC, RSI, RSO and OV flag. If the result of the last arithmetic or logical operation is zero, the Z flag will be set. If the result of the last arithmetic or logical operation is negative, the N flag will be set. Control instructions use the N and Z flags to determine if a jump is required or not.

3.3.3 Data Pointer (DPX)

MCS 51 microcontrollers have a 16-bit DPTR that can access up to 64-Kbyte range of data memory. The MCS 251 microcontrollers have a 24-bit wide extended data pointer, DPX, that can provide a full 16-Mbyte of data memory access. The reset value for DPX is 010000H. External data fetch using instruction via the DPTR, will fetch data from region 01: of the 8XC251SB memory space. The 8XC251SB can fetch external data within the 64 Kbyte region using any of the word registers, WRj, or outside the 64 Kbyte region using any of the double-word registers, DRk. Note that the content of DPX is also reflected in the double-word register DR56.

3.3.4 Stack Pointer (SPX)

The 8XC251SB has a 16-bit wide stack pointer, SPX, compared to MCS 51 microcontrollers that have an 8-bit stack pointer, SP. Hence, the 8XC251SB provides 65,536 bytes (64 Kbytes) of stack space compared to MCS 51 microcontrollers which have only 256 bytes of stack space. This large amount of stack space is advantageous especially in the high-level language programming where data gets pushed or popped to/from the stack during a function or routine call. New instructions, PUSH Rm, PUSH WRj, PUSH DRk, POP Rm, POP WRj and POP DRk, can move data of different byte sizes on or off the stack very quickly. Note that the 8XC251SB INTR bit in the Configuration Register can be programmed for a 2 or 4 bytes. During the interrupt service routine with the INTR-bit set for 4-byte, three bytes of PC and one byte of PSW1 will be pushed (or popped) onto (or off) the stack. Note that the content of SPX is also reflected in the double-word register DR60.

4.0 ADDRESSING MODES

The 8XC251SB has seven groups of addressing modes: register addressing, immediate addressing, direct addressing, indirect addressing, indirect displacement addressing, relative addressing and bit addressing. Each group of these addressing modes may have more than one type of addressing modes. In the next few sections, 8XC251SB addressing modes will be discussed.

The 8XC251SB has seven types of addressing modes that are not available in the MCS® 51 microcontrollers. Refer to Table 2 for a brief comparison of addressing mode available in MCS 51 micrccontrollers and the 8XC251SB.

Table 2. Addressing Modes in MCS 51 and 8XC251SB Microcontrollers

Addressing Mode	MCS® 51 Microcontrollers	8XC251SB
	X	X
Word register addressing	—	X
Double-word register addressing	—	X
8-bit direct addressing	X	X
16-bit direct addressing	—	X
8-bit immediate addressing	X	X
16-bit immediate addressing	—	X
8-bit indirect addressing	X	X
16-bit indirect addressing	X	X
24-bit indirect addressing	—	X
16-bit displacement addressing	—	X
24-bit displacement addressing	—	X
Relative addressing	X	X
Bit addressing	X	X

4.1 Register Addressing

In the section on Register File, it was mentioned that high-level language compilers use registers to store variables for computation of an arithmetic expression. These instructions operate using Register Addressing mode. Register addressing is where the instruction specifies the register(s) that contains the operand. MCS 51 microcontrollers have only one type of register addressing mode: byte register (Rn). The 8XC251SB has three types of register addressing mode: byte(s) register (Rn) addressing, word(s) register (WRj) addressing or double-word(s) register (Drk) addressing. 8XC251SB register addressing mode has register to register operation, which is not supported in the MCS 51 microcontrollers. 8XC251SB register to register operation will provide smaller code size but bigger if the same task were to be performed by the MCS® 51 microcontrollers. In addition to code size reduction, it also will reduce execution time, hence this will increase the overall system performance.

Examples of instructions that uses the register addressing that are not available in MCS 51 microcontrollers:

```
ADD     R0, R1              ; R0=R0 + R1
MOV     R0, R1              ; R0=R1
SUB     WR0, WR2            ; WR0=WR0 - WR2
MUL     WR0. WR2            ; WR0=WR0 x WR2
```

4.2 Immediate Addressing

Immediate addressing is where the instruction contains the operand. MCS 51 microcontroller supports only 8-bit immediate data (#dat). 8XC251SB has 8-bit immediate data (#dat) and 16-bit immediate data (#dat16). Examples of instructions that uses the immediate addressing that is not available in MCS 51 microcontrollers:

```
ADD     WR0, #2000H         ; WR0=WR0 + #2000H
ORL     WR0, #05555H        ; WR0= WR0 OR #05555H
MOV     WR0, #05555H        ; WR0=#05555H
PUSH    #01234H             ; [SP]=12H, [SP+1]=34H, SP=SP+2
```

4.3 Direct Addressing

Direct addressing is where the instruction contains the operand address. MCS 51 microcontroller supports 8-bit direct addressing of the on-chip RAM location 00H to 7FH (128bytes) and SFRs location 80H to FFH.

The 8XC251SB can perform both 8-bit and 16-bit direct addressing. Refer to Figure 1 for direct addressing in the memory address space. It also provides direct addressing to the SFRs location S:80H to S:FFH. The 8XC251SB provides 64 Kbytes that can be directly accessed compared to MCS 51 microcontrollers that provide only 128 bytes. Examples of instructions that use the direct addressing that is not available in MCS 51 microcontrollers:

```
MOV     WR0. 00:0200H       ; R0=[00:0200H] & R1=[00:0201H]
ADD     R0, 00:0200H        ; R0=R0 + [00:0200H]
ADD     WR0. 00:0200H       ; WR0=WR0 + [0200H, 0201H]
MOV     DR0, 00:0200H       ; R0=[00:0200H], R1=[00:0201H]
                            ; R2=[00:0202H], R3=[00:0203H]
```

4.4 Indirect Addressing

Indirect addressing is where the instruction specifies the register that contains the operand address. MCS 51 indirect addressing supports 8-bit addresses using registers, R0 and R1 only, and 16-bit addresses using data pointer, DPTR. The 8XC251SB, indirect addressing supports 8-bit, 16-bit and 24-bit addresses. Refer to Figure 1 for indirect addressing in the memory address space. 16-bit indirect addressing is not limited to DPTR, Wrj registers can also be used. For a 24-bit indirect addressing, DRk must be used. Examples of instructions that use the indirect addressing that is not available in MCS 51 microcontrollers:

```
ADD     R0, @WR8        ; R0=R0 + [[WR8]]
MOV     @DR0, WR0       ; [[DR0]] = WR0
ORL     R0, @DR8        ; R0= R0 OR [[DR8]]
LJMP    @WR0            ; PC= [[WR0]]
ECALL   @DR0            ; PC=[[DR0]]
```

4.5 Displacement Addressing

Indirect displacement addressing is where the instruction specifies a register and an offset. This addressing scheme is not available in the MCS 51 microcontroller but is available for MCS 251 data transfer instructions. These instructions transfer bytes or words of data within the 16-Mbyte memory address space. Example of instructions that uses the indirect displacement addressing that is not available in MCS 51 microcontrollers:

```
MOV     R0, @WR8 + 2000H        ; R0= [[WR8]+ #2000H]
MOV     WR0, @WR8 + 2000H       ; R0= [[WR8] + #2000H],
                                ; R1=[[WR8] + #2001H]
MOV     @WR4 + 2000H, WR0       ; [[WR4]+ #2000H]=WR0
MOV     @DR4 + 2000H, R0        ; [[DR4]+ #2000H]= R0
```

4.6 Relative Addressing

Relative addressing: The instruction contains an 8-bit signed offset from the next instruction to the target address. Relative addressing is also available for 8XC251SB. New instructions such as JE, JNE, JG, JLE, JSL, JSLE, JSG and JSGE use this relative addressing mode.

```
JG      LABEL1          ; LABEL1 is a relative address
JSG     LABEL2          ; LABEL2 is a relative address
```

4.7 Bit Addressing

Bit addressing is where the instructions contain the bit address. MCS 51 microcontrollers have only 16 bytes of on-chip RAM and 16 bytes of SFRs that are bit addressable, whereas the 8XC251SB has 96 bytes of on-chip RAM and all 128 bytes of SFR locations that are bit addressable. Refer to Figure 1 for bit addressing in the memory address space. Examples of instructions that use the bit addressing that is not available in MCS 51 microcontrollers:

```
CLR     S:7FH.0         ; Clear 7FH bit 0
JNB     S:099H.7, LABEL ; IF SBUF bit 7 not set THEN jump to LABEL
```

5.0 INSTRUCTION SET

The instruction set of the MCS 51 microcontrollers is a subset of the instruction set found in the 8XC251SB. 8XC251SB supports all the 255 instructions found in the MCS 51 microcontrollers and many more new instructions. Instructions found in MCS 51 microcontroller will not be discussed here but they are available in the *MCS 51 Microcontroller Family User's Manual* and only new types of instructions will be discussed. These instructions can be grouped into the following instruction sets: arithmetic instructions, logical instructions, data transfer instructions, bit instructions and control instructions. In the next few sections, some of the new types of instructions will be discussed. A detailed description of these instructions is available in the *8XC251SB User's Manual.*

5.1 Arithmetic Instructions

In the MCS 51 microcontrollers, the arithmetic instructions operate only on byte data and must be done through the accumulator. For applications that require intensive arithmetic operation, this becomes a bottleneck. The 8XC251SB enhanced instruction set has register to register operations, this overcome the performance bottleneck. The 8XC251SB provides byte, word and double-word operations to further enhance computation of an arithmetic functions. These operations are done through Rm and WRj registers. Refer to Appendix A for a list of arithmetic instructions that compares CPU cycle between MCS 51 microcontrollers and 8XC251SB.

Refer to Appendix B for examples of using new 8XC251SB arithmetic instructions

5.2 Data Transfer Instructions

Most microcontroller-based systems spend a large amount of their CPU cycles (time) moving data from one location to the other. If there was a way to decrease the CPU cycle (time) to transfer the data, then the overall performance of the system could be improved. 8XC251SB has a very rich set of MOV instructions that allow data to be moved either as byte, word or double-word.

Example 3a shows how MCS 51 microcontrollers move data from external data memory to register R0 using an indirect addressing via the accumulator pointed by DPTR. The content in the accumulator is then moved to the register R0.

```
; Using MCS 51 microcontrollers to move data from external data memory to a
; register
          mov       DPTR, #data-addr                    ; DPTR=data_addr
          clr       A                                   ; A=00H
          movx      A, @A+DPTR                          ; A=[[A+DPTR]]
          mov       R0, A                               ; R0=A
; Total number of MCS 51 microcontroller states= 30
; Total number of MCS 51 microcontroller bytes= 7
```

Example 3a. Using MCS 51 Microcontrollers for a Data Move

Example 3b shows one of the many new MOV instructions provided by the 8XC251SB which bypasses the DPTR. Taking advantage of the register based architecture, the 8XC251SB is able to indirectly move the data to the register R0.

```
; Using 8XC251SB to move data from external data memory to a register
          mov       WR2, #data-addr                     ; WR2=#data_addr
          mov       R0. @WR2                            ; R0=[[WR2]]
; Total number of 8XC251SB states= 4
; Total number of 8XC251SB bytes= 7
```

Example 3b. Using the 8XC251SB for a Data Move

Example 3a and 3b show that the 8XC251SB executes 7.5 times faster than the MCS 51 microcontrollers for the same code size.

Example 4a illustrates how MCS 51 microcontrollers move 2 bytes of data stored sequentially in the on-chip data RAM to registers for an arithmetic operation or other user operation.

```
; MCS 51 microcontrollers to move 2 bytes of data stored sequentially in the
; on-chip data RAM to registers
        mov       R0, addr_data_1                     ; R0=addr_data_1
        mov       R1, addr_data_2                     ; R1=addr_data_2
        |
        Arithmetic operation or other user operation
        |
; Total number of MCS 51 microcontroller states= 24
; Total number of MCS 51 microcontroller bytes= 4
```

Example 4a. Another Example of the Using MCS 51 Microcontrollersfor Data Move

Example 4b shows how the 8XC251SB moves data stored in the on-chip general-purpose RAM for arithmetic operation or other user operation. The 8XC251SB has reduced the execution time from 24 states to only 3 states.

```
; 8XC251SB to move 2 bytes of data stored sequentially in the on-chip data RAM
; to registers
        mov       WR0, addr_data_1      ; R0=[addr_data_1] & R1=[addr_data_2]
        |
        Arithmetic operation or other user operation
        |
; Total number of 8XC251SB states= 3
; Total number of 8XC251SB bytes= 3
```

Example 4b. Another Example of Using 8XC251SB for Data Move

Example 4a and 4b show that the 8XC251SB executes 8 times faster than the MCS 51 microcontrollers. Also there is a 13% code size reduction using the 8XC251SB.

Refer to Appendix A for a list of data transfer instructions that compares CPU cycle between MCS 51 microcontrollers and the 8XC251SB.

Refer to Appendix C for more examples that use the 8XC251SB data transfer instructions.

5.3 Logical Instructions

MCS 51 microcontrollers logical instructions can only operate on 8-bit data. 8XC251SB logical instruction supports 8-bit and 16-bit data operation. Register to register operation can be performed on the 8XC251SB but not on the MCS 51 microcontrollers. By using the Rm and WRj registers to resolve the accumulator based bottle-neck of the MCS 51 microcontrollers. overall system performance is enhanced.

Example 5a illustrate how MCS 51 microcontrollers logically rotate the content of RO register. Firstly, the content of RO is moved to the accumulator. Accumulator is then rotated. Finally, the result is moved back to RO register.

```
; MCS 51 microcontrollers to logically rotate the content of RO register.
        mov     A, RO                   ; A=[RO]
        rl      A                       ; Rotate accumulator to the left
        mov     RO, A                   ; RO=A
; Total number of MCS 51 microcontroller states= 18
: Total number of MCS 51 microcontroller bytes= 3
```

Example 5a. MCS 51 Microcontrollers Logically Rotate the Content of the RO Register

Example 5b illustrates how the 8XC251SB can directly rotate the content of RO register without using the accumulator.

```
; 8XC251SB directly gate RO
        sll                             : Rotate RO to the left
; Total number of 8XC251SB states= 1
; Total number of 8XC251SB bytes= 2
```

Example 5b. 8XC251SB Logically Rotate the Content of the RO Register

Example 5a and 5b show that the 8XC251SB executes 18 times faster than the MCS 51 microcontrollers. Also there is a 15% code size reduction using the 8XC251SB.

Refer to Appendix A for a list of logical instructions that compares the CPU cycle between MCS 51 microcontrollers and 8XC251SB.

Refer to Appendix D for more examples that use the 8XC251SB logical instructions.

5.4 Bit Instructions

MCS 51 microcontrollers bit instructions operate on 16 bytes of the internal data memory address space and 16 bytes of the SFR space. In comparison, the 8XC251SB bit instructions operate on a larger memory region, that is, 96 bytes of the memory address space and 128 bytes of the SFR space. With the increase in the bit addressable space, this allows more Boolean variables and SFRs registers to be controlled directly.

Refer to Appendix A for a list of bit instructions that compares CPU cycle between MCS 51 microcontrollers and 8XC251SB.

5.5 Control Instructions

8XC251SB offers 15 new control instructions. Some of these new instructions, for example ECALL, ERET and EJMP, allow the user to take advantage of the entire 8XC251SB memory space. High-level languages such as C can take advantage of these new control instructions, JE, JNE, JG, JLE. JSL, JSLE, JSG and JSGE together with arithmetic instructions such as SUB, ADD, CMP and etc.

Refer to Appendix A for a list of control instructions that compares CPU cycle between MCS 51 microcontrollers and 8XC25ISB.

Refer to Appendix E for examples that use the 8XC251SB logical instructions.

6.0 CONCLUSION

This application note coveres the programming aspects of 8XC251SB and some software performance comparison between the MCS 51 microcontrollers and 8XC251SB. It shows how the 8XC251SB executes much faster than the MCS 51 microcontrollers. It also illustrates how the 8XC251SB requires smaller code space versus MCS 51 micro-controllers. This application note also shows that the 8XC251SB have higher performance, an increased memory mix and addressing, efficient high-level language support, enhanced instructions set to the 8-bit embedded microcon-troller market available today.

APPENDIX A: INSTRUCTION CYCLE IMPROVEMENT

INSTRUCTION SET SUMMARY

ARITHMETIC OPERATIONS			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
ADD	A,Rn	6	2
ADD	A,direct	6	1
ADD	A,@Ri	6	2
ADD	A,#data	6	1
ADD	Rm,Rm	—	1
ADD	WRj,WRj	—	2
ADD	Rm,#data	—	2
ADD	WRj,#data16	—	3
ADD	Rm,dir	—	2
ADD	WRj,dir	—	3
ADD	Rm,dirx	—	2
ADD	WRj,dirx	—	3
ADD	Rm,@WRj	—	2
ADD	Rm,@DRk	—	3
ADDC	A,Rn	6	2
ADDC	A,direct	6	1
ADDC	A,@Ri	6	2
ADDC	A,#data	6	1
SUB	Rm,Rm	—	1
SUB	WRj,WRj	—	2
SUB	Rm,#data	—	2
SUB	WRj,#data16	—	3
SUB	Rm,dir	—	2
SUB	WRj,dir	—	3
SUB	Rm,dirx	—	2
SUB	WRj,dirx	—	3

ARITHMETIC OPERATIONS			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
SUB	Rm,@WRj	—	2
SUB	Rm,@DRk	—	3
SUBB	A,Rn	6	2
SUBB	A,direct	6	1
SUBB	A,@Ri	6	2
SUBB	A,#data	6	1
INC	A	6	1
INC	Rn	6	2
INC	direct	6	2
INC	@Ri	6	3
INC	Rm, #short	—	1
INC	WRj, #short	—	1
DEC	A	6	1
DEC	Rn	6	2
DEC	direct	6	2
DEC	@Ri	6	3
DEC	Rm, #short	—	1
DEC	WRj, #short	—	1
INC	DPTR	12	1
MUL	AB	24	5
MUL	Rm,Rm	—	5
MUL	WRj,WRj	—	11
DIV	AB	24	10
DIV	Rm,Rm	—	10
DIV	WRj,WRj	—	20
DA	A	6	1

LOGICAL OPERATIONS

Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
ANL	A,Rn	6	2
ANL	A,direct	6	1
ANL	A,@Ri	6	2
ANL	A,#data	6	1
ANL	direct,A	6	2
ANL	direct,#data	12	2
ANL	Rm,Rm	—	1
ANL	WRj,WRj	—	2
ANL	Rm,#data	—	2
ANL	WRj,#data16	—	3
ANL	Rm,dir	—	2
ANL	WRj,dir	—	3
ANL	Rm,dirx	—	2
ANL	WRj,dirx	—	3
ANL	Rm,@WRj	—	2
ANL	Rm,@DRk	—	3
ORL	A,Rn	6	2
ORL	A,direct	6	1
ORL	A,@Ri	6	2
ORL	A,#data	6	1
ORL	direct,A	6	2
ORL	direct,#data	12	2
ORL	Rm,Rm	—	1
ORL	WRj,WRj	—	2
ORL	Rm,#data	—	2
ORL	WRj,#data16	—	3
ORL	Rm,dir	—	2
ORL	WRj,dir	—	3
ORL	Rm,dirx	—	2
ORL	WRj,dirx		3
ORL	Rm,@WRj	—	2
ORL	Rm,@DRk	—	3
XRL	A,Rn	6	2
XRL	A,direct	6	1
XRL	A,@Ri	6	2
XRL	A,#data	6	1
XRL	direct,A	6	2
XRL	direct,#data	12	2
XRL	Rm,Rm	—	1
XRL	WRj,WRj	—	2
XRL	Rm,#data	—	2
XRL	WRj,#data16	—	3
XRL	Rm,dir	—	2
XRL	WRj,dir	—	3
XRL	Rm,dirx	—	2
XRL	WRj,dirx	—	3
XRL	Rm,@WRj	—	2
XRL	Rm,@DRk	—	3
CLR	A	6	1
CPL	A	6	1
RL	A	6	1
RL	Rm	—	1
RL	WRj	—	1
RLC	A	6	1
RLC	Rm	—	1
RLC	WRj	—	1
RR	A	6	1
RR	Rm	—	1
RR	WRj	—	1
RRC	A	6	1
RRC	Rm	—	1
RRC	WRj	—	1
SLL	Rm	—	1
SLL	WRj	—	1
SRA	Rm	—	1
SRA	WRj	—	1
SRL	Rm	—	
SRL	WRj	—	1
SWAP	A	6	

DATA TRANSFER			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
MOV	A,Rn	6	2
MOV	A,direct	6	1
MOV	A,@Ri	6	2
MOV	A,#data	6	1
MOV	Rn,A	6	2
	Rn,direct	12	2
	a	6	2
MOV	direct,A	6	2
MOV	direct,Rn	12	3
MOV	direct,direct	12	3
MOV	direct,@Ri	12	3
MOV	direct,#data	12	2
MOV	@Ri,A	6	3
MOV	@Ri,direct	12	3
MOV	@Ri,#data	6	3
MOV	DPTR,#data16	12	2
MOV	Rm,Rm	—	1
MOV	WRj,WRj	—	1
MOV	Rm,#data	—	2
MOV	WRj,#data16	—	3
MOV	Rm,dir	—	2
MOV	WRj,dir	—	3
	Rm,dirx	—	2
		—	3
MOV	Rm,@WRj	—	2
MOV	Rm,@DRk	—	3
MOV	dir, Rm	—	2
MOV	dir, WRj	—	3
MOV	dirx, Rm	—	2
MOV	dirx, WRj	—	3

DATA TRANSFER			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
MOV	@WRj, Rm	—	3
MOV	@DRk, Rm	—	4
MOV	Rm,@WRj+dis	—	5
MOV	WRj,@WRj+dis	—	6
MOV	Rm,@DRk+dis	—	6
MOV	WRj,@DRk+dis	—	7
MOV	@WRj+dis, Rm	—	6
MOV	@WRj+dis, WRj	—	8
MOV	@DRk+dis, Rm	—	7
MOV	@DRk+dis, WRj	—	9
MOVC	A,@A+DPTR	12	5
MOVC	A,@A+PC	12	6
MOVX	A,@Ri	12	4
MOVX	A,@DPTR	12	4
MOVX	@DPTR,A		
PUSH	direct B/W	12	5
PUSH	#data	12	5
PUSH	#data16	—	3
PUSH	dirx B/W	—	3
PUSH	Rm	—	3
PUSH	WRj	—	5
POP	direct	12	3
POP	Rm	—	2
POP	WRj	—	5
POP	dirx	—	3
XCH	A,Rn	6	3
XCH	A,direct	6	3
XCH	A,@Ri	6	4
XCHD	A,@Ri	6	4

BOOLEAN VARIABLE MANIPULATION			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
CLR	C	6	1
CLR	bit	6	2
CLR	bit_dir	—	3
CLR	bit_dirx	—	4
SETB	C	6	1
SETB	bit	6	2
SETB	bit_dir	—	3
SETB	bit_dirx	—	4
CPL	C	6	1
CPL	bit	—	2
CPL	bit_dir	—	3
CPL	bit_dirx	—	4
ANL	C,bit	12	1
ANL	C, bitdir	—	2
ANL	C, bitdirx	—	3
ANL/	C,/bit	12	1
ANL/	C, /bit_dir	—	2
ANL/	C, /bit_dirx	—	3
ORL	C,bit	12	1
ORL	C, bit_dir	—	2
ORL	C, bit_dirx	—	3
ORL/	C,/bit	12	1
ORL/	C, /bit_dir	—	2
ORLI	C, /bit_dirx	—	3
MOV	C,bit	6	1
MOV	C, bit_dir	—	2
MOV	C, bit_dirx	—	3
MOV	bit,C	12	2
MOV	bit_dir, C	—	3
MOV	bit_dirx, C	—	4
JC	rel	12	1
JNC	rel	12	1

BOOLEAN VARIABLE MANIPULATION			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
JB	bit,rel	12	2
JB	bit,relx	—	3
JNB	bit,rel	12	2
JNB	bit,relx	—	3
JBC	bit,rel	12	2
JBC	bit,relx	—	3

PROGRAM BRANCHING			
Mnemonic	Note	MCS® 51 Controller States	8XC251SB States*1
ACALL	addr11	12	6
ECALL	addr24	—	12
ECALL	@DRk	—	12
LCALL	addr16	12	6
ERET		—	8
RET		12	6
RETI		12	8
AJMP	addr11	12	3
EJMP	addr24	—	4
EJMP	@DRk	—	5
LJMP	addr16	12	4
SJMP	rel	12	3
JMP	@A + DPTR	12	4
JZ	rel	12	2
JNZ	rel	12	2
CJNE	A,direct,rel	12	3
CJNE	A, #data,rel	12	2
CJNE	Rn, #data,rel	12	3
CJNE	@Ri, #data,rel	12	3
DJNZ	Rn,rel	12	3
DJNZ	direct,rel	12	3
NOP		6	1

NOTE:
***1: The 8XC251SB execute from internal code memory and the pipeline is full.**

APPENDIX B: EXAMPLES OF NEW 8XC251SB ARITHMETIC INSTRUCTION

```
; Example 1:
; Description: Perform 8 bit data addition using registers.
; R0=R0+R1.
;***************************************************************************

;    ***MCS® 51 Microcontroller***
          mov      A, R0
          add      A, R1                ; A=R0+R1
          mov      R0. A                ; R0=A
; Total number of MCS 51 microcontroller states=18
; Total number of MCS 51 microcontroller bytes= 3

;    ***8XC251SB microcontroller***
          add      R0, R1               ; R0=R0+R1
; Total number of 8XC251SB microcontroller states= 1
; Total number of 8XC251SB microcontroller bytes= 2
```

```
; Example 2:
; Description: Perform 16 bit data addition using registers.
; [R0,R1]=#1234H+#5678H.
;***************************************************************************
;    ***MCS 51 microcontroller***
          mov      R1, #78H             ; R1=#78H
          mov      R0, #56H             ; R0=#56H
          mov      A, #34H              ; A=#34H
          add      A, R1                ; A=#34H+78H
          mov      R1, A                ; R1=A
          mov      A, #12H              ; A=#12H
          addc     A, R0                ; A=#12H+#56H
          mov      R0, A                ; R0=A
; Total number of MCS 51 microcontroller states= 54
; Total number of MCS 51 microcontroller bytes= 12

;    ***8XC251SB microcontroller***
          Using ADD WRj, WRj
          mov      WR0, #1234H          ; Operand 1
          mov      WR2, #5678H          ; Operand 2
          add      WR0, WR2             ; WR0=#1234H + #5678H
; Total number of 8XC251SB microcontroller states= 6
; Total number of 8XC251SB microcontroller bytes= 10
; Example 3:
; Description: Double-word addition.
```

```
; Double-word register= #12345678H + #87654321H
;*****************************************************************************

;    *** MCS 51 Microcontroller ***
          mov       A. #78H          ; A=#78H
          mov       R3, #21H         ; R3=#21H
          add       A, R3            ; A=A+RO
          mov       R3, A            ; R3=A
          mov       A, #56H          ; A=#56H
          mov       R2, #43H         ; R2=#43H
          addc      A, RO            ; A=A+R2
          mov       R2, A            ; R2=A
          mov       A, #34H          ; A=#34H
          mov       Rl, #65H         ; Rl=56H
          addc      A, Rl            ; A=A+Rl
          mov       Rl, A;           ; Rl=A
          mov       A, #12H          ; A=#12H
          mov       RO, #87H         ; RO=#87
          addc      A, RO            ; A=A+RO
          mov       RO. A            ; RO=A
; Total number of MCS® 51 Microcontroller states= 102
; Total number of MCS 51 Microcontroller bytes = 24

;    *** 8XC251SB ***
          Using ADD DRJ,DRJ instructions to perform 4 bytes of addition
          operation.
          mov       WRO. #1234H      ; WRO=#1234H
          mov       WR2, #1234H      ; WR2=#5678H
          mov       WR4, #4321H      ; WR4=#8765H
          mov       WR6, #4321H      ; WR6=#4321H
          add       DRO, DR4         ; WRO=WRO+WR2
; Total number of 8XC251SB states=12
; Total number of 8XC251SB bytes=18
```

```
; Example 4:
; Description: Perform 8 bit data multiplication
; RO=RO x Rl
;*****************************************************************************
;    ***MCS 51 microcontroller***
          mov       A, #12H          ; A=#12H
          mov       B. #34H          ; B=#34H
          mul       AB               ; A x B
          mov       Rl, A            ; Rl=Low byte result
          mov       RO, B            ; RO=High byte result
; Total number of MCS 51 microcontroller states= 60
; Total number of MCS 51 microcontroller bytes= 9

;    ***8XC251SB microcontroller***
          mov       WRO, #1234H      ; WRO=#1234H
          mul       RO, Rl           ; WRO= RO x Rl
; Total number of 8XC251SB microcontroller states= 7
; Total number of 8XC251SB microcontroller bytes= 6
; Example 5:
; Description: Multiply 2 16-bit numbers and store in registers.
;******************************************************************************
          Number1(Number1+1)
;    X    Number2(Number2+1)
;    ----------------------------
          R0    R1    R2    R3
          R0=Most significant byte of double-word result
          R3=Least significant byte of double-word result
;    ***MCS 51 Microcontroller***
          (Number1+1) x (Number2+1)
          mov      R0, #00H             ; R0=#00H
          mov      R1, #00H             : R1=#00H
          mov      B, (Number1+1)       ; B=(Number1+1)
          mov      A, (Number2+1)       ; A=(Number2+1)
          mul      AB                   ; A x B
          mov      R3, A                ; R3=A
          mov      R2, B                ; R2=B
          (Number1) x (Number2+1)
          mov      B, Number1           ; B=(Number1)
          mov      A, (Number2+1)       ; A=(Number2+1)
          mul      AB                   ; A x B
          add      A, R2                ; A=A+R2
          mov      R2, A                : R2=A
          mov      A, B                 ; A=B
          addc     A. R1                ; A=A+R1
          mov      R1, A                ; R1=A
          jnc      Mul_loop1            ; IF Carry=0 THEN jump
          inc      R0                   ; R0=R0+1
Mul__loop1:
          (Number1+1) x (Number2)
          mov      B, (Number1+1)       ; B=
          mov      A, Number2           ; A=Number2
          mul      AB                   ; A x B
          add      A, R2                : A=A+R2
          mov      R2, A                ; R2=A
          mov      A. B                 ; A=B
          addc     A, R1                ; A=A+R1
          mov      R1, A                ; R1=A
          jnc      Mul_loop2            ; IF Carry=0 THEN jump
          inc      R0                   ; R0=R0+1
Mul__loop2:
          (Number1) x (Number2)
          mov      B, Number1           ; B=Number1
          mov      A, Number2           ; A=Number2
          mul      AB                   ; A x B
          add      A, R1                ; A=A+R1
          mov      R1, A                : R1=A
          mov      A, B                 ; A=B
          addc     A. R0                ; A=A+R0
          mov      R0, A                ; R0=A
; Total number of MCS® 51 microcontroller states= 312
; Total number of MCS 51 microcontroller bytes= 55
;    ***8XC251SB microcontroller***
,         (Numberl, Numberl+1) X (Number2, Number2+1)


          mov      WRO, Numberl        ; Operand 1
          mov      WR2, Number2        ; Operand 2
          mul      WRO, WR2            ; DRO= WRO x WR2
; Total number of 8XC251SB microcontroller states= 17
; Total number of 8XC251SB microcontroller bytes= 10
```

```
; Example 6:
; Description: Increment RO by 4
;*****************************************************************************
;    ***MCS 51 microcontroller***
          mov      A, RO
          add      A, #04H             ; Add RO with 04H
          mov      RO, A
; Total number of MCS 51 microcontroller states= 18
; Total number of MCS® 51 microcontroller bytes= 4

;    ***8XC251SB microcontroller***
          inc      RO, #04H            ; Increment RO by 4 times
; Total number of 8XC251SB microcontroller states= 1
; Total number of 8XC251SB microcontroller bytes= 2
```

```
; Example 7:
; Description: Decrement RO by 4
;*****************************************************************************
;    ***MCS 51 microcontroller***
          mov      A. RO               ; A=RO
          subb     A, #04H             ; A=A--04H
          mov      RO. A               ; RO=A
; Total number of MCS 51 microcontroller states= 18
; Total number of MCS 51 microcontroller bytes = 4

;    ***8XC251SB microcontroller***
          dec      RO, #04H            ; RO=RO--04H
; Total number of 8XC251SB microcontroller states= 1
; Total number of 8XC251SB microcontroller bytes= 2
; Example 8:
; Description: Compare data in RO and R1 and jump accordingly.
;*****************************************************************************
;     *** MCS 51 Microcontroller ***
          mov       A, RO                  ; A=RO
          cjne      A, 01H, NOT__EQUAL     ; If RO <> R1 Then
EQUAL:
; Total number of MCS 51 Microcontroller states= 18
; Total number of MCS 51 Microcontroller bytes= 4


;     *** 8XC251SB ***
          Using JNE and CMP instructions.
          cmp       RO, R1                 ; If RO ? R1
          jne       NOT--EQUAL             ; If RO<>R1 Then Jump
EQUAL:
; Total number of 8XC251SB states= 2
; Total number of 8XC251SB bytes= 4
```

APPENDIX C: EXAMPLES OF NEW 8XC251SB DATA TRANSFER INSTRUCTION

```
; Example 1:
; Description: Copy bytes from register to register.
; Data word store in R0 (High-byte) and R1 (Low-byte] move to R4 and R5.
;****************************************************************************
;    *** MCS® 51 Microcontroller ***
         There is no instruction for register to register operation, hence
         use direct addressing.
         mov      04H, 00H           ; R4=R0
         mov      05H, 01H           ; R5=R1
; Total number of MCS 51 Microcontroller states= 24
; Total number of MCS 51 Microcontroller bytes=6

;    *** 8XC251SB ***
         mov      WR4, WR0           ; R4=R0 & R5=R1
; Total number of 8XC251SB cycles= 1
; Total number of 8XC251SB bytes = 2
```

```
; Example 2:
; Description: Move a word data from external memory to internal data memory.
;****************************************************************************
;    *** MCS 51 Microcontroller ***
         ; Setup pointer
         mov      DPL. (SRC+1)       ; DPL=(SRC+1)
         mov      DPH, SRC           ; DPL=SRC
         ; Get data & Store data
         movx     A, -DPTR           ; A=[[DPTR]]
         mov      DST, A             ; DST=A
         inc      DPTR               ; DPTR=DPTR+1
         movx     A, -DPTR           ; A=[[DPTR]]
         mov      DST+1, A           ; (DST+1)=A
; Total number of MCS 51 Microcontroller states= 72
; Total number of MCS 51 Microcontroller bytes= 24

;    *** 8XC251SB ***
         mov      WR0, #SRC          ; WR0=SRC
         mov      WR2, -WR0          ; WR2=[[WR0]]
         mov      DST, WR2           ; DST=WR2
; Total number of 8XC251SB states= 10
; Total number of 8XC251SB bytes= 9
; Example 3:
; Description: Move one byte of data from code memory to register.
;*************************************************************************
;    *** MCS® 51 Microcontroller ***
           ; Setup pointer
          mov  DPL. (SRC+1)        ; DPL=(SRC+1)
          mov  DPH, SRC            ; DPH=SRC
           ; Get and store data byte
          clr  A                   ; A=00H
          movc A, @A+DPTR          ; A=[[A+DPTR]]
          mov  R0, A               ; R0=A
; Total number of MCS 51 Microcontroller states= 48
; Total number of MCS 51 Microcontroller bytes= 9

;    *** 8XC251SB ***
           ; Setup pointer
          mov  WR2. SRC            ; WR2=SRC
           ; Get and store data byte
          mov  R0, @WR2            ; R0=[[WR2]]
; Total number of 8XC251SB states= 5
; Total number of 8XC251SB bytes= 6
```

```
; Example 4:
; Description: Moving 32 bytes of data from code memory location starting at
; SRC0 to data memory starting at location DST_BLK
;*************************************************************************
;    *** MCS® 51 Microcontroller ***
XSEG      AT   0100H
DST_BLK:       DS     1FH

CSEG      AT   0000H
          ljmp MAIN

CSEG      AT   0100H
SRC0:     DB   0H,1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
SRC10:    DB   10H,11H,12H,13H,14H,15H,16H,17H,18H,19H,1AH,1BH,1CH,1DH,1EH,1FH
TOTAL:    DB   20H

;**************************************************************
BLOCK-MOVE:
           ; Get TOTAL
          mov  DPL, #LOW(TOTAL)     ; DPL=LOW(TOTAL)
          mov  DPH, #HIGH(TOTAL)    ; DPH=HIGH(TOTAL)
          clr  A                    ; A=00H
          movc A. @A+DPTR           ; A=[[A+DPTR]]
          mov  R0. A                ; R0=A
           ; Get DST address
          mov  R4, #LOW(DST_BLK)    ; R4=LOW(DST_BLK)
          mov  R5, #HIGH(DST_BLK)   ; R5=HIGH(DST_BLK)
           ; Get SRC addr
          mov  DPL, #LOW(SRC0)      ; DPL=LOW(SRC0)
          mov  DPH. #HIGH(SRC0)     ; DPL=HIGH(SRC)
```

```
MOVE:
          ; Get data
          clr  A                ; A=0
          movc A, -A+DPTR       ; A=[[A+DPTR]]
          mov  R2, DPL          ; R2=DPL
          mov  R3, DPH          ; R3=DPH
          ; Store data to DST
          mov  DPL, R4          ; DPL=R4
          mov  DPH. R5          ; DPH=R5
          movx -DPTR, A         ; [[DPTR]]=A
          ; Point to next DST
          inc  DPTR             ; DPTR=DPTR+1
          mov  R4, DPL          ; R4=DPL
          mov  R5. DPH          ; R5=DPH
          ; Point to next SRC
          mov  DPL. R2          ; DPL=R2
          mov  DPH, R3          ; DPH=R3
          inc  DPTR             ; DPTR=DPTR+1
          djnz R0, MOVE         ; IF R0<>0 THEN jmp to MOVE
.****************************************************************
; Total number of MCS 51 Microcontroller states= 5262
; Total number of MCS 51 Microcontroller bytes= 42
          ret
MAIN:
          acall BLOCK-MOVE
END

;****************************************************************
;    *** 8XC251SB ***
DSEG      AT   00:0100H
DST_BLK:       DS1     FH

CSEG      AT   FF:0000H
          ljmp MAIN

CSEG      AT   FF:0100H
SRC0:     DB   0H,1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
SRC10:    DB   10H,11H,12H,13H,14H,15H,16H,17H,18H,19H,1AH,1BH,1CH,1DH,1EH,1FH
TOTAL:    DB   10H

.****************************************************************
BLOCK-MOVE:
          ; Get TOTAL
          mov  WR24, #00FFH; WR24=#00FFH
          mov  WR26, #LOW16(TOTAL); DPTR=#TOTAL
          mov  R0,@DR24         ; R0=[[DR24]]
          ; Get DSTaddress
          mov  WR4, #DST_BLK    ; WR4=#DST_BLK
          ; Get SRC addr
          mov  WR26, #LOW16(SRC0); WR26=#LOW(SRC0)
```

```
MOVE:
          ; Get data


          mov      WR2, @DR24          ; WR2=[[DR24]]
          ; Store to DST
          mov      @WR4, WR2           ; [[WR4]]=WR2
          ; Point to next SRC & DST
          inc      WR26, #2            ; WR26=WR26+2
          inc      WR4, #2             ; WR4=WR4+2
          djnz     R0, MOVE            ; IF R0<>0 THEN jump to MOVE
;****************************************************************
; Total number of 8XC251SB states= 263
; Total number of 8XC251SB bytes= 33
          ret
MAIN:
          acall    MOVE
END
```

APPENDIX D: EXAMPLES OF NEW 8XC251SB LOGICAL INSTRUCTION

```
; Example 1:
; Description: Logical AND 2 bytes.
;*****************************************************************
;     *** MCS® 51 Microcontroller ***
          mov       A. RO                ; A=RO
          anl       A, Rl                ; A=A AND Rl
          mov       Rl, A                : Rl=A
; Total number of MCS 51 Microcontroller states= 18
; Total number of MCS 51 Microcontroller bytes= 3

;     *** 8XC251SB ***
          anl       RO, Rl               ; Logical AND bytes
; Total number of 8XC251SB states= 1
; Total number of 8XC251SB bytes= 2
```

```
; Example 2:
; Description: Logical AND byte data using indirect addressing.
•*****************************************************************
;     *** MCS 51 Microcontroller ***
          Setup pointer
          mov       DPL, (SRC+1)         ; DPL=(SRC+1)
          mov       DPH, SRC             ; DPH=SRC
          Get code data
          clr       A                    ; A=OOH
          movc      A. @A+DPTR           ; A=[[A+DPTR]]
          Perform logical AND
          anl       A. Pl                ; A=A ANL Pl
          Store data
          mov       R3, A                ; R3=A
; Total number of MCS 51 Microcontroller state= 54
; Total number of MCS 51 Microcontroller bytes= 13

;     *** 8XC251SB ***
          Setup pointer
          mov       WRO, SRC             ; WRO=#SRC
          Perform logical AND
          mov       R2, Pl               ; R2=Pl
          anl       R2, @WRO             ; R2 =R2 AND [[WRO]]
          Store data
          mov       P3, R2               ; Move result to port3
; Total number of 8XC251SB states= 10
; Total number of 8XC251SB bytes= 13
; Example 3:
; Description: Logical AND word data
; [R0R1] ANL [R2R3]
;********************************************************************
;    *** MCS 51 Microcontroller ***
         mov      A, R0                ; A=R0
         anl      A, R2                ; A= A ANL R2
         mov      R0, A                ; R0=A
         mov      A. R1                ; A=R1
         anl      A, R3                ; A=A ANL R3
         mov      R1, A                ; Move low result to R1
; Total number of MCS 51 Microcontroller states= 36
; Total number of MCS 51 Microcontroller bytes= 8

;    *** 8XC251SB ***
         anl      WR0, WR2             ; WR0=WR0 ANL WR2
; Total number of 8XC251SB states=2
; Total number of 8XC251SB bytes= 2
```

APPENDIX E: EXAMPLES OF NEW 8XC251SB CONTROL INSTRUCTION

```
; Example 1:
; Description: Compare word and jump accordingly.
;*******************************************************************
;     4** MCS® 51 Microcontroller ***
          Compare high byte
          mov       RO, SRC               : RO= SRC
          cjne      RO, #7FH, NOT-EQUAL ; If RO <> #7FH Then
H_EQUAL:
          Compare low byte
          mov       RO, (SRC+1)           ; RO=(SRC+1)
          cjne      RO, #OFFH, NOT-EQUAL; If RO<>#FFH Then
EQUAL:
          ajmp      EQUAL
NOT-EQUAL:
          jc        LESS                  ; If SRC <#7FFFH Then
          ajmp      GREATER               ; If SRC > #7FFFH
; Total number of MCS 51 Microcontroller states= 84
; Total number of MCS 51 Microcontroller bytes= 18

;     *** 8XC251SB ***
          Using JE. JLE and CMP Wrj, #data16 instructions.
          mov       WRO, SRC              ; WRO=SRC
          cmp       WRO, #7FFFH           ; If WRO ? #7FFFH
          je        EQUAL                 ; If RO=R1 Then
          jle       LESS                  ; If SRC < #7FFFH Then
          ajmp      GREATER               : If SRC > #7FFFH Then
; Total number of 8XC251SB states= 8
; Total number of 8XC251SB bytes=11
; Example 2:
: Description: Show EJMP, ECALL and ERET instructions
•*****************************************************************************
:     *** 8XC251SB ***
; Reset vector
ORG        FE:0000H
           ejmp     MAIN
           . . .

; Region FE:
ORG        FE:0100H
SUB1:
           . . .
           eret

; Region FF:
ORG        FF:0100H
MAIN:
           . . .
           ecall SUB1
           . . .
END
```

ADDITIONAL REFERENCES

Detailed information on the 8XC251SB device funtionality and product specifications can be obtained in the following literature:

- *8XC251SB User's Manual* (Order Number 272617)
- *8XC251SB High—Performance CHMOS Single Chip Microcontroller* datasheet (Order Number 272616)
- AP-708, Introducing the MCS® 251 Microcontroller—8XC251SB (Order Number 272670)
- AP-710, Migrating from the MCS® Microcontroller to **the** MCS 251 Microcontroller—Software and Hardware Considerations (Order Number 272672)

INTEL CORPORATION, 2200 Mission College Blvd., Santa Clara, CA 95052; Tel. (408) 765-8080

INTEL CORPORATION (U.K.) Ltd., Swindon, United Kingdom; Tel. (0793) 696 000

INTEL JAPAN k.k., Ibaraki-ken; Tel. 029747-8511

Printed in USA/EBM95-077.0/395 5K/SP DS